Raport z przygotowania projektu na inżynierię lingwistyczną

Julia PoteraLskA

przygotowanie i Preprocessing danych

Na początku zainstalowałam narzędzia oraz zaimportowałam biblioteki, które będą niezbędne do wykonania kodu (transformers, datasets, wandb etc.). Następnie wczytałam treningowe zbiory danych z komputera (z opiniami i ocenami), dzieląc je na zbiory danych z całymi opiniami i zbiory danych z poszczególnymi zdaniami (ds\_reviews\_sentences, ds\_reviews, ds\_emotions\_sentences, ds\_emotions).

Moim kolejnym krokiem było wczytanie dwóch datasetów testowych. Je także podzieliłam na zbiory danych z całymi opiniami i zbiory danych z poszczególnymi zdaniami (ds\_testA, ds\_testB, ds\_testA\_sentences, ds\_testB\_sentences). Sprawdziłam liczność danych, a także, czy występują braki w danych lub puste stringi. Nie było takich przypadków.

Następnie przystąpiłam do przygotowania danych do przetwarzania przez transformery. Na początku zajęłam się całymi zdaniami. Stworzyłam listę słowników, w której każdemu wierszowi tekstu ze zbioru treningowego przyporządkowałam listę emocji, z wartością 1 dla True i 0 dla False. Podzieliłam powstały zbiór na dataset treningowy i walidacyjny. Przekształciłam ramki danych z danymi na datasety Hugging Face.

Stworzyłam ‘final\_dataset‘, który jest słownikiem powyższych datasetów. Do zbioru danych dotrenuję model typu BERT („distilbert-base-uncased”). Wczytuję tokenizator dla modelu (AutoTokenizer, który automatycznie wybiera odpowiedni tokenizator na podstawie nazwy modelu). Tworzę funkcję tokenizującą tekst i za jej pomocą tokenizuję zbiór danych. Pobieram pretrenowany model z Hugging Face i określam, ile jest docelowych klas.

Do pojedynczych zdań chciałam użyć innego modelu typu BERT („bert-base-cased”), ponieważ zachowuje on wielkość liter, co może być istotne przy analizie większego zbioru danych, dostarczając bardziej precyzyjnych wyników. Jednak ostatecznie zdecydowałam się na distilberta, ponieważ jest on mniej wymagający pamięciowo. Procesy w tej części są podobne jak w przypadku opisanym powyżej.

Dodaję funkcje metryki: F1 macro score: .

Definiuje parametry treningu, od ich wartości zależy, jak dobre wyniki uzyskamy. Następnie trenuje modele.

Ewaluuje model przy użyciu final score: .

Na tym etapie napotkałam wiele problemów związanych z treningiem i wandb, m.in. zużycie pamięci RAM czy brak łączności z układem GPU z powodu limitów wykorzystania obowiązujących w Colab. Starałam się je rozwiązywać. Po wielu próbach udało mi się je zlikwidować.

Podsumowanie

Zalety, wady i ograniczenia

**Z technicznego punktu widzenia, automatyzacja procesów przetwarzania danych i treningu modeli przyspiesza proces. Wykorzystanie bibliotek umożliwia efektywną obsługę dużych zbiorów danych dzięki zaawansowanej tokenizacji i batchowaniu. Integracja z wandb pozwala na śledzenie wyników eksperymentów w czasie rzeczywistym oraz na dostrajanie hiperparametrów (co w tej pracy akurat pominęłam), co zwiększa elastyczność i efektywność pracy nad modelem. Dodatkowo, zarządzanie eksperymentami za pomocą wandb umożliwia efektywne porównywanie różnych podejść i wybór najlepszych konfiguracji modelu.**

**Projekt nie jest pozbawiony wad. Technicznie, wymaga on instalacji i zarządzania wieloma zewnętrznymi bibliotekami, co może być kłopotliwe. Ponadto, trening modeli wymaga znacznych zasobów obliczeniowych, co jest kosztowne i czasochłonne. Kod można by było skrócić, aby uniknąć powtórzeń i zachować przejrzystość. Oprócz tego, projekt wymaga dostrojenia hiperparametrów.**

**Przyjęty sposób przetwarzania zakłada specyficzne formaty, co jest problematyczne w przypadku różnych struktur danych. Pretrenowane modele mogą również zawierać uprzedzenia, które wpływają na wyniki i interpretację emocji, co jest znaczącą wadą, zwłaszcza w kontekście etyki i modelowania.**

**Zauważyłam, że występują też pewne ograniczenia modelu. Przetwarzanie danych zakłada specyficzne formaty, przez co struktura danych używana w projekcie ogranicza elastyczność projektu. Model może nie generalizować dobrze do danych znacząco różniących się od zbioru treningowego, co ogranicza jego uniwersalność i zastosowanie w różnych kontekstach. Dodatkowo, potencjalne uprzedzenia w pretrenowanych modelach mogą wpływać na interpretację wyników, prowadząc do błędnych wniosków i decyzji.**

M**ożliwości ulepszeń w przyszłości**

**Aby projekt mógł być bardziej efektywny i uniwersalny, warto wprowadzić techniki augmentacji danych, które mogą zwiększyć odporność modelu na różnorodne dane i poprawić jakość danych wejściowych.**

**Wykorzystanie metod ensemble, czyli łączenia predykcji z wielu modeli, może poprawić dokładność wyników.   
Szerokie dostrajanie hiperparametrów pozwoli na optymalizację wydajności modelu (na przykład poprzez zastosowanie grid searchu).**

**Podsumowując, mimo pewnych wad i ograniczeń, projekt ma solidne podstawy, które można jeszcze bardziej wzmocnić i wykorzystać w przyszłości do wypracowania lepiej działającego i ulepszonego projektu.**